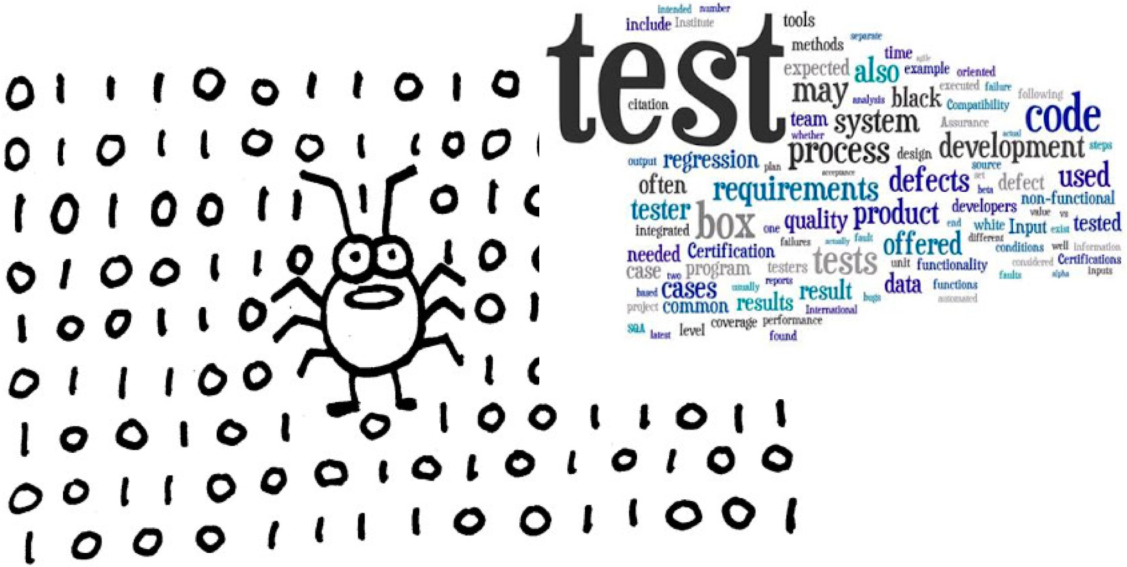
Towards Automatic Inference of Inductive Invariants

Haojun Ma, Aman Goel, Jean-Baptiste Jeannin
Manos Kapritsos, Baris Kasikci, Karem A. Sakallah

University of Michigan

Distributed systems are subtle



The alternative: formal verification

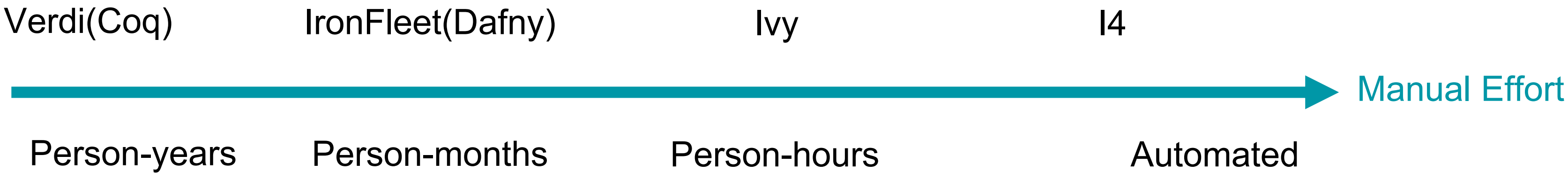
Formal specification or property

Proving the system maintains the property

Successful on distributed systems

Drawback: Manual effort

Existing verification approaches



Verdi(Coq)	IronFleet(Dafny)	Ivy	I4	
Person-years	Person-months	Person-hours	Automated	Manual Effort

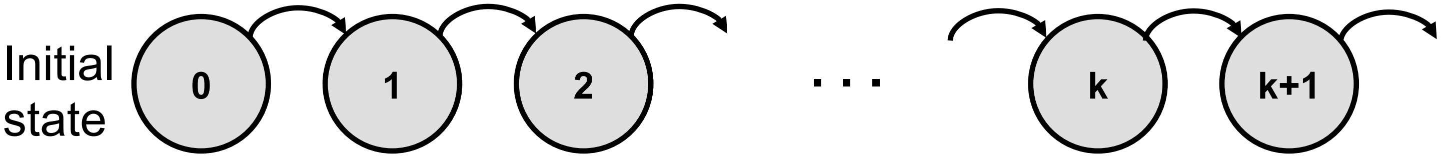
All existing approaches require the human to find an **inductive invariant**

We want to automatically find inductive invariants

Formal verification in 2 minutes

Goal: prove that the safety property holds **at all times**

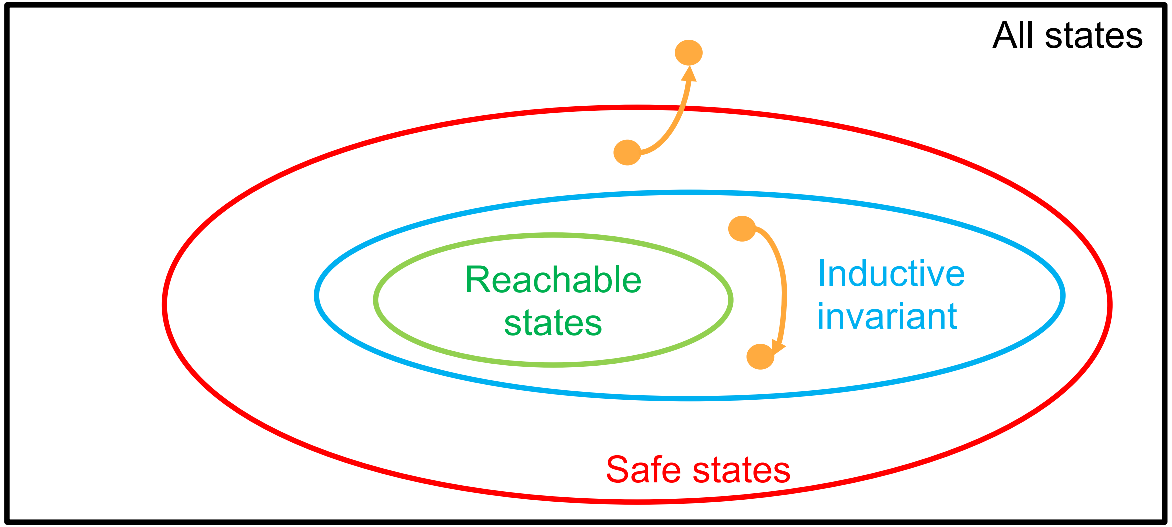
An execution:



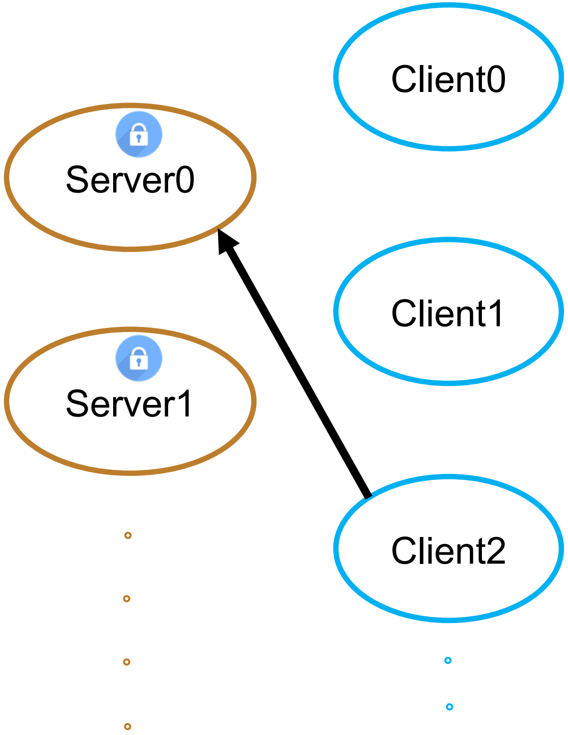
Inductive proof

- Base case: prove initial state is safe
- Inductive step: if state **k** is safe, prove state **k+1** is safe

Safety property vs. inductive invariant



Lock server protocol



Safety property:

no two clients can be linked to the same server

$$\forall C_0, C_1, S.\ link(C_0, S) \wedge link(C_1, S) \implies C_0 = C_1$$

Finding an inductive invariant using Ivy

Automatically checks if an invariant is inductive

(Screenshot from Ivy)

Requires **t**

Existing approaches rely on manual effort and human intuition

$$\forall C_0, C_1, S.\ link(C_0, S) \wedge link(C_1, S) \implies C_0 = C_1$$ Safety property

$$\wedge \qquad \forall C, S.\ link(C, S) \implies \neg lock\_hold(S)$$

ion

Outline

Motivation

I4: a new approach

Design of I4

Evaluation

Future work

I4: a new approach

Goal: Find an inductive invariant ***without*** relying on human intuition.

Insight: Distributed protocols exhibit ***regularity***.

- Behavior doesn’t fundamentally change as the size increases
- E.g. lock server, Paxos, …

Implication: We can use inductive invariants from small instances to infer a ***generalized*** inductive invariant that holds for all instances.

Leveraging model checking

Model checking

- ☺ Fully automated
- ☹ Doesn’t scale to distributed systems

I4 applies model checking to small, finite instances …

… and then generalizes the result to all instances.

Outline

Motivation

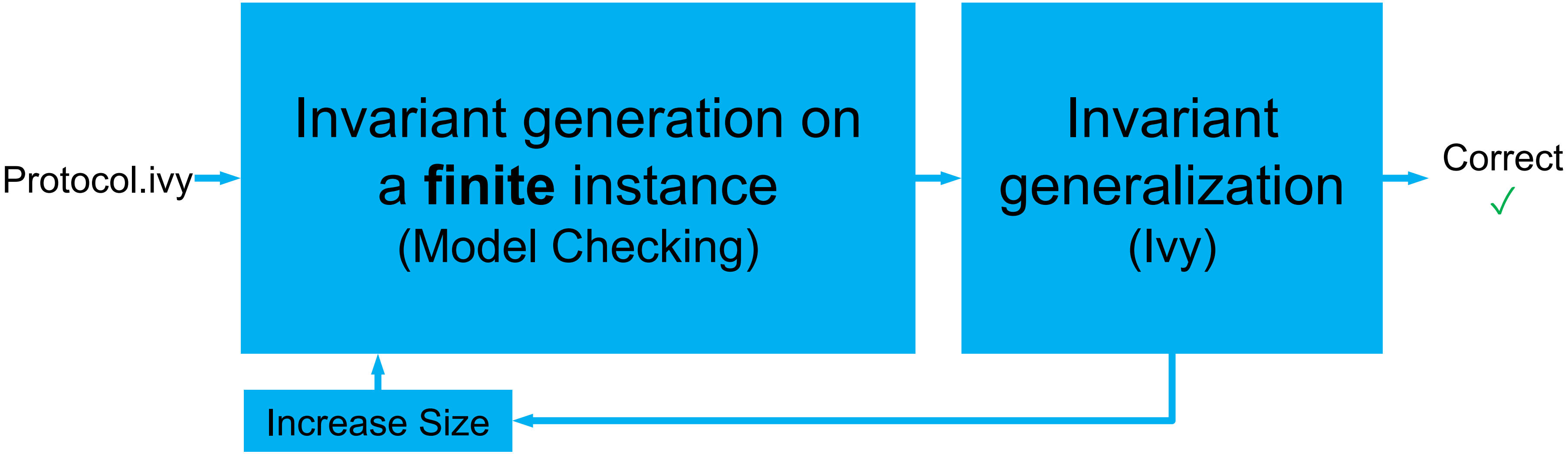
I4: a new approach

Design of I4

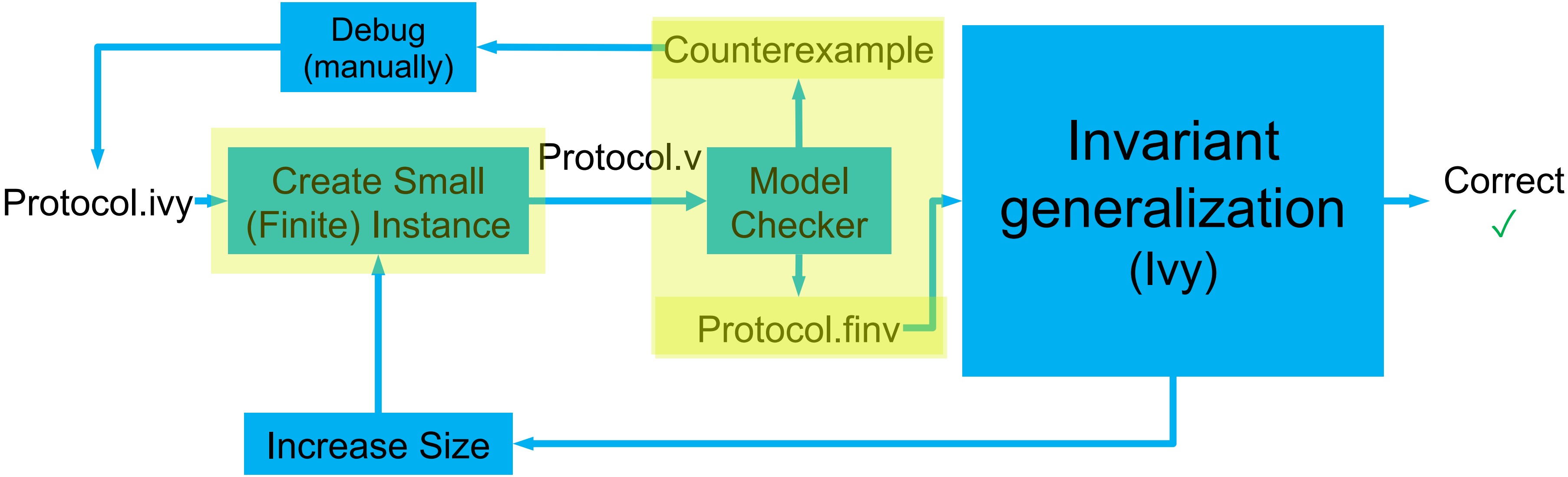
Evaluation

Future work

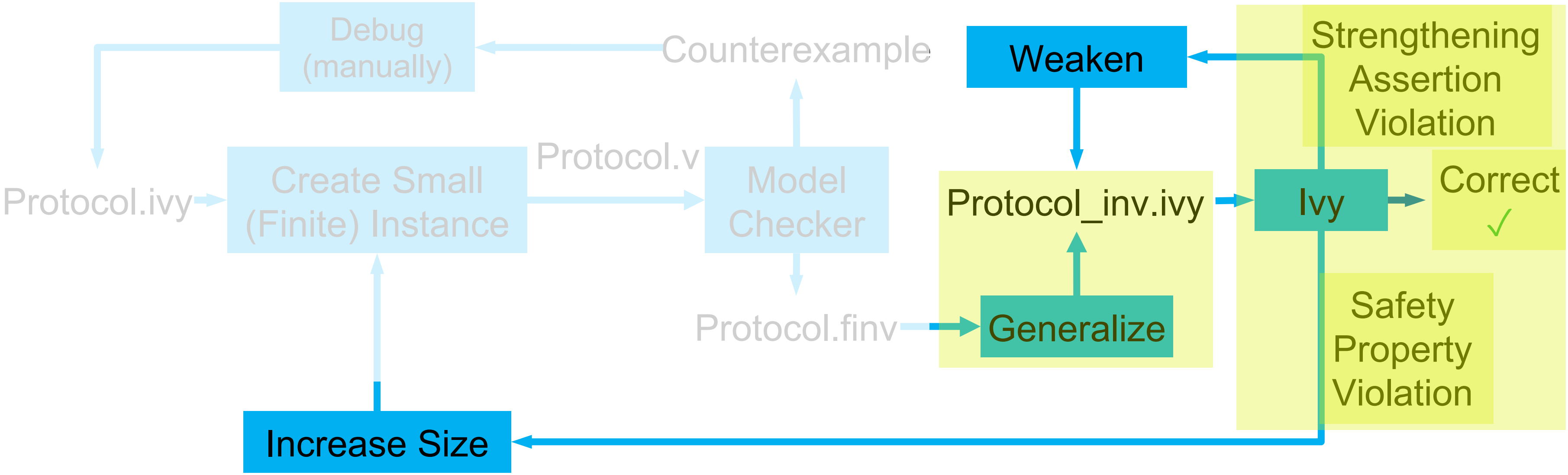
Overview


Protocol.ivy
Invariant generation on a **finite** instance
(Model Checking)
Invariant generalization
(Ivy)
Correct
✓
Increase Size

Invariant generation on a finite instance



Invariant Generalization



Outline

Motivation

I4: a new approach

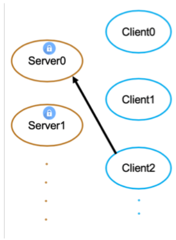
Design of I4

Evaluation

Future work

Evaluation

Lock Server

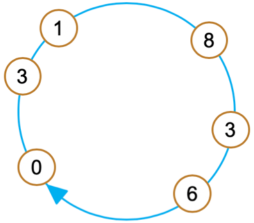


1 server
2 clients

~3s

✓

Leader Election

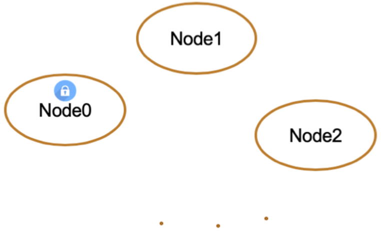


3 nodes
3 IDs

~8s

✓

Distributed lock



2 nodes
4 epochs

~12s

✓

Outline

Motivation

I4: a new approach

Design of I4

Evaluation

Future work

Future work

More automation

Scalability to larger protocols

Verification of Implementations